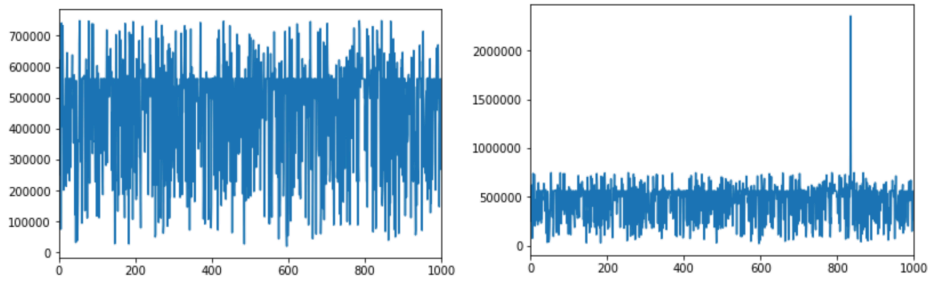
Summary: This summarizes a benchmark experiment comparing the impact of instance type and quantity on convolutional neural net learning performance and speed for pet classification.

Research design: Accuracy and elapsed time records are kept as 8 combinations of image sample quantity and quality are classified as either 'cat' or 'dog': **(1)** quantity of sample data is tested at 2000 and 4000 images total: for the latter, all image instances are horizontally flipped and appended to original 2000 image data sets. **(2)** quality is varied into 4 cells: 64 vs 128 bit and grey scale (1 channel) vs color (2 channel RGB). **(3)** Neural net layers start with two convolutional: 1st with 32 and 2nd with 64 filters aka feature maps, 1st with strides = 1 and 2nd with strides = 2 and ksize = 3 for both. **(4)** One reason this paper selects sample augmentation is that many different techniques were tried, but in all cases validation accuracy could not rise above 65-69% and in some cases test and validation accuracy was very jumpy. **(5)** One technique, **local response normalization** treats the output of both convolutional layers with radius 2m, alpha = 0.0002, and beta = 0.75. The intent is to polarize neuron confidence so that some are strongly activated and simultaneously inhibit collocated neurons in other filters, resulting in specialization. These are employed in the 1st two layers to collect larger features for upper layer to act upon. (6) a fourth layer employs a 2x2 **max pooling filter** to reduce parameter numbers and thus overfitting and encourage tolerance for image shift (which was trained by flipping pets horizontally). (7) The output of the subsequent **fully connected** neural net layer is treated with **batch normalization** to scale and (8) is followed by an output logits layer. (9) **Forward pass activation** is via rectified linear unit aka ReLU in the convolutional layers and elu in the fully connected. (10) **Selu activation** was tested in the convolutional layers but produced jumpy test accuracy. (11) Time permitting, I would add experiments for relu vs selu

in convolutional layers and elu versus the version of selu that coexists with batch normalization. (12) An **AdamOptimizer** minimizes **sparse_softmax_cross_entropy_with_logits** mean value.

Data exploration / preparation: (1) Python library opencv converts 1000 dog and cat images from original form to all 8 combinations of 512, 256, 128, and 64 -bit and 1 and 3-channels. (2) Numpy flattens these images into arrays, and (3) via its 'fliplr' function, numpy flips their duplicates across a vertical axis and appends to the original array to provide more samples for our learning vehicle. (4) Along the way it is noticed that original image 835 (shown here) has well over 2 million pixels **(LHS below)** versus all other photos ranging 100-700 k pixels.



Python code: Identical hardware and libraries are employed for these as for Assignement 6. (1) MiniMaxScaler normalizes the (2) concatenated cat and dog features sets. (3) Labels for the y dataset are manually created for cats (zeros) and dogs (ones). (4) 80/20 train/test splits are randomly executed via sklearn library and 70/10 train/validation subsets via indexation. (5) Parameters for filter count, ksize, stride and padding are assigned as variables outside tensorflow. (6) A flattened X is assigned to a placeholder tensor for mini batch gradient and (7) a reshaped X tensor is assigned and fed to the first convolutional which like other layers has zero padded filters. (8) Local response normalization, then convolutional, then local response normalization are followed by max pooling, and fully connected which is batch normalized. (9) Learning rate was dumbed down to 0.001 to calm learning which was very jumpy as SELU and

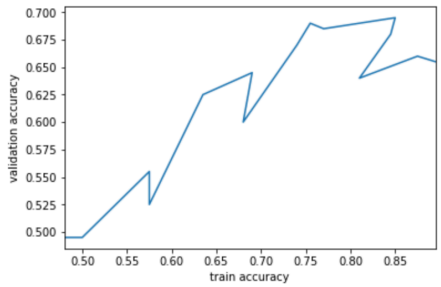
other parameters were loosely experimented with before formal testing began. (10 The above code developed in modular form, but wrapped in a function and sprinkled with list gatherings in order to form metrics for the experiments that are timed by Python's time.process_time().

Table 1: Treatments, Speed, Accuracy, Dimensions

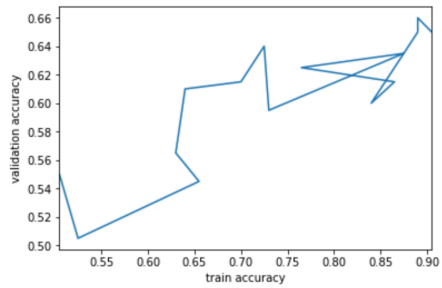
	treatment	time	acc train	acc test	images	pixels	channels	conv1	conv2	pool3	fc	logits
0	2000_64_1	11550.919824	0.867500	0.69250	2000	64	1	(?, 64, 64, 32)	(?, 32, 32, 64)	(?, 16, 16, 64)	(?, 64)	(?, 2)
1	2000_64_3	12729.581956	0.898125	0.71500	2000	64	3	(?, 64, 64, 32)	(?, 32, 32, 64)	(?, 16, 16, 64)	(?, 64)	(?, 2)
2	2000_128_1	50398.597228	0.950625	0.67500	2000	128	1	(?, 128, 128, 32)	(?, 64, 64, 64)	(?, 32, 32, 64)	(?, 64)	(?, 2)
3	2000_128_3	55273.239346	0.951875	0.64500	2000	128	3	(?, 128, 128, 32)	(?, 64, 64, 64)	(?, 32, 32, 64)	(?, 64)	(?, 2)
4	4000_64_1	23692.840290	0.895312	0.71375	4000	64	1	(?, 64, 64, 32)	(?, 32, 32, 64)	(?, 16, 16, 64)	(?, 64)	(?, 2)
5	4000_64_3	27010.372230	0.918750	0.72875	4000	64	3	(?, 64, 64, 32)	(?, 32, 32, 64)	(?, 16, 16, 64)	(?, 64)	(?, 2)
6	4000_128_1	93882.974328	0.961875	0.69750	4000	128	1	(?, 128, 128, 32)	(?, 64, 64, 64)	(?, 32, 32, 64)	(?, 64)	(?, 2)
7	4000_128_3	114984.158904	0.934687	0.69375	4000	128	3	(?, 128, 128, 32)	(?, 64, 64, 64)	(?, 32, 32, 64)	(?, 64)	(?, 2)

Results: Results shown in **table** 1 above demonstrate the highest test **accuracy** when the sample size is doubled for 64 bit images whether grey scale or color as shown in row 4 and 5 in table 1. Higher train accuracy scores are available in all other rows but generalization is even worse in rows 0-3 and 6-7 than in rows 4 and 5. Generalization can be seen from the **plots below** as train accuracy is plotted on the horizontal and validity accuracy on the vertical axis. The numerical identifiers for each plot can be associated with **"treatment"** in table 1 above. **Elapsed time** doubles with each step in the experiment from rows (0,1) to rows (4,5) to rows (2,3) and finally to rows(6,7), but the sweet spot appears to be rows 4 and 5.

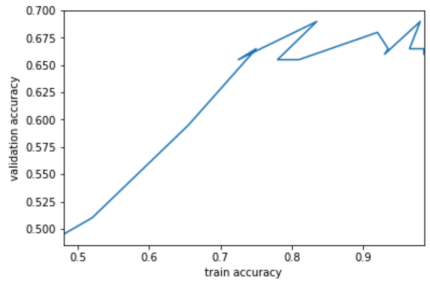
2000_64_1



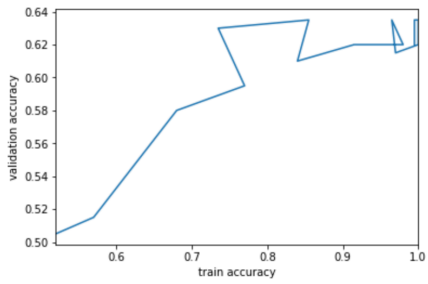
2000_64_3



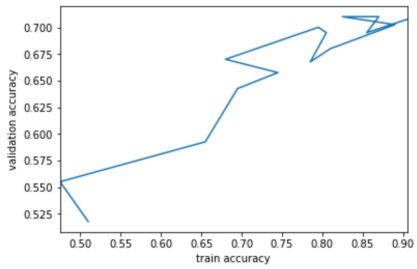
2000_128_1



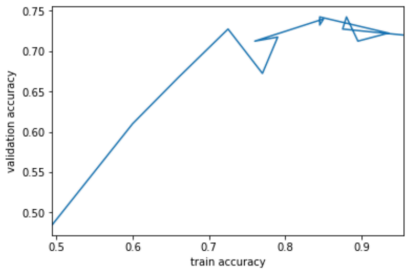
2000_128_3



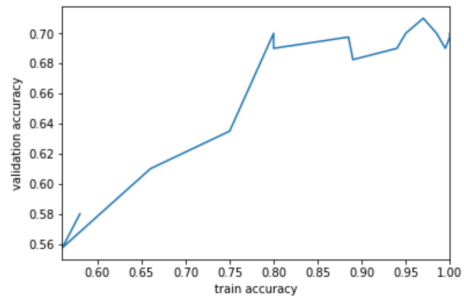
4000_64_1



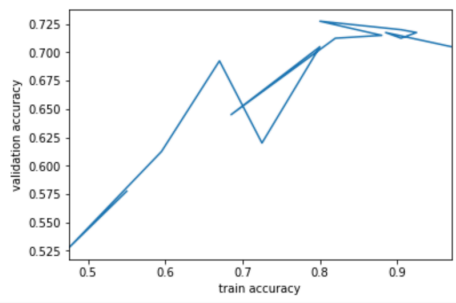
4000_64_3



4000_128_1



4000_128_3



Management recommendations. Based on these results, and managements expressed disregard for elapsed time, pursuing image augmentation for larger sample size is recommended. Prior to the formal testing shown above, varying batch sizes and iterations,

different activation methods (elu, selu), and drop out were tried with either no improvement in validation accuracy past 70% or dramatically more volatile train accuracy, bouncing from 40 to 90%. More work can be with these parameters, but the most well-behaved train accuracy and improved validation accuracy came from augmenting the sample. I would recommend continuing with that line of analysis. If image shift is explored further, then max pooling should be retained and varied until optimized.